Available online at www.sciencedirect.com

ScienceDirect

Procedia Computer Science 36 (2014) 387 – 392

Procedia Computer Science

Complex Adaptive Systems, Publication 4
Cihan H. Dagli, Editor in Chief
Conference Organized by Missouri University of Science and Technology
2014-Philadelphia, PA

Towards a Compiler for a Polychronous Wavefront Computer: Programming by Optimization

Corey B. Hart*

Lockheed Martin IS&GS, 230 Mall Blvd, King of Prussia, PA, 19406, USA

Abstract

The incorporation of delay based computing, or polychronicity, into models of neural networks has helped to increase the memory and representational capacity of spiking neural networks. However, the computational advantages of spiking neural networks are largely obviated if they are instantiated on a conventional computer architecture. An alternative architecture that has been advanced is the paradigm of polychronous wavefront computation (PWC). PWC is a framework in which transponders embedded in some kind of wave-conductive medium are used as a simplified representation of computational processes similar to those exhibited by spiking neural networks. Programming such a network amounts to determining the proper geometrical arrangement of transponders within the conductive medium. With this in mind we therefore conjecture that transforming a mathematical function into the corresponding PWC geometry (i.e., compiling the representational code for that function) could best be done by some form of swarm-based optimization within the physical space of potential geometries. We herein test the ability of a swarm algorithms (particle swarm optimization) to select arrangements capable of encoding simple mathematical functions and compare the convergence times against one another as well as against optimization algorithms with less obvious geometrical interpretations (genetic algorithms).

Peer-review under responsibility of scientific committee of Missouri University of Science and Technology

Keywords: neural networks, non-von Neumann computation, parallel processing, computational architectures, biomimetic, theory

1. Introduction

Nomenclature

PWC	Polychronic Wavefront Computation
PSO	Particle Swarm Optimization
GA	Genetic Algorithm

*Corresponding author *E-mail address*: corey.hart@lmco.com

Peer-review under responsibility of scientific committee of Missouri University of Science and Technology
doi:10.1016/j.procs.2014.09.010

1.1. Background

While digital computation has made great strides in terms of computational power in recent decades, there are some indications that this modality is approaching a natural limit. As such, there has been significant interest of late in the exploration and

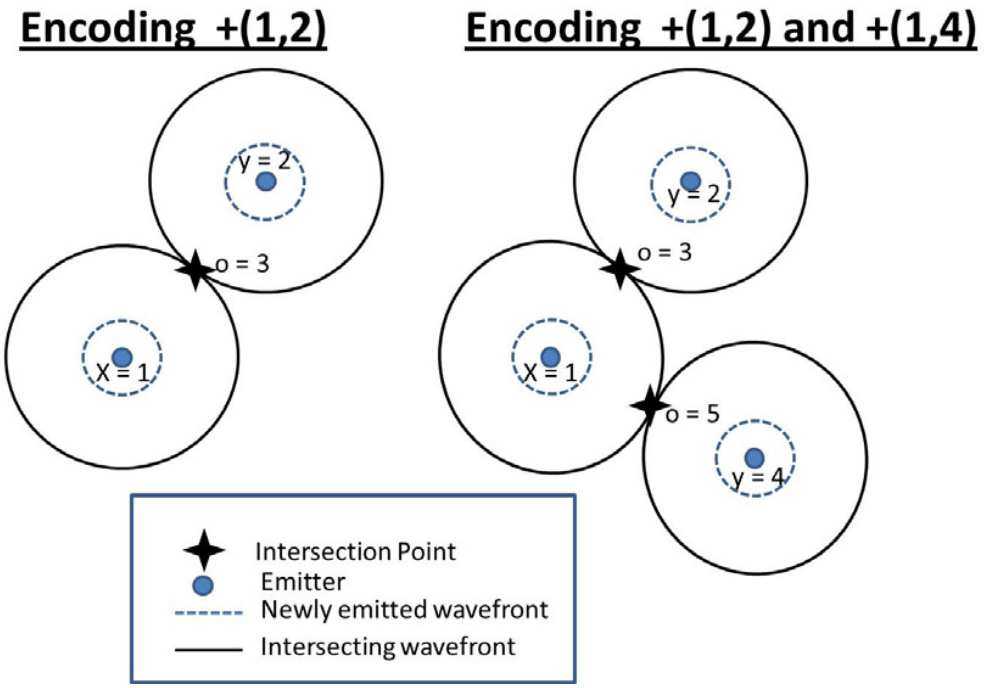

Figure 1: Basic schema for encoding binary adders into a polychromous wavefront computer. Inputs x and y for the function +(x,y) are assigned a particular position in a medium. When stimulated, they emit waves that intersect at particular regions of the workspace. Transponders acting as readouts must then lie along these intersections.

development of alternative computational substrates that may possess significant computational advantages over conventional structures. Many different architectures for novel computer hardware have been explored or implemented, ranging from the esoteric, such as quantum computing [1] or optical computers[2,3] to the pragmatic, best exemplified by so-called approximate math processors[4].

One of the more interesting developments in non-digitial computation has been the adoption of architectures inspired by neural network structures. Clearly the brain is a powerful computer, and it is only rational to try and draw inspiration from its architecture. This has lead to a proliferation of neuromorphic processors[5,6] that recapitulate this structure to a greater or lesser degree. The most realistic of these architectures take inspiration from what is known as the third generation of neural networks. These networks consist of spiking neurons connected with both realistic time-delays and learnable synaptic weights. Because spikes propagate along axons and reach target synapses of particular strengths after a time delay, relations between neurons (and therefore between elements of a memory) may be encoded by either the delay line structure between neurons or the strength of the synaptic connection between them, or some combination of these. This variability in encoding increases the memory and computational capacity of the network[7] over that exhibited by conventional neural networks, which use only synaptic connection strength for memory encoding. While spike processing hardware is currently under development, the recognition that both time delay and synaptic strength contribute to the encoding of information in

spike processing hardware has led to a more general proposed architecture: the Polychronous Wavefront Computer[8] a computer architecture in which time delays and exchanges between transponders in some medium represent the entirety of the computation being performed. It is similar to a neuromorphic architecture in that information is transmitted by discrete exchanges between emitters, but differs in that information is not limited to propagating along axons or other analogues to physical delay lines. Instead, emitted signals propagate uniformly into some space and excitations of new emitters occur when two or more propagating signals impinge on those emitters. In effect, the geometry of the transponders represents the computer program instantiated by the architecture. Because a polychronous wavefront computer is well poised to take advantages of the efficiencies[9,10] implicit in optical computing as well as those offered by delay based computing, we herein explore a simple technique for programming relations in PWC hardware using a particle swarm optimization (PSO). We expected this to be a successful technique for programming because of the mapping between geometry and programming in PWC. As a simple test of this hypothesis, we compared the time-to-converge for PSO-based program compilation and a genetic algorithm implementation of the same.

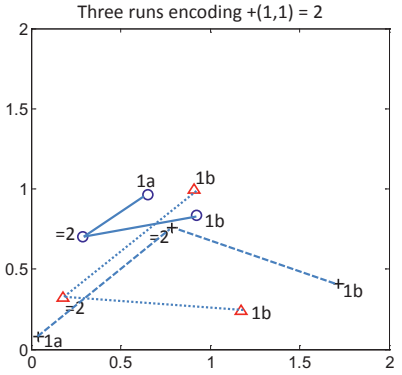

Figure 2: Encoding +(1,1) = 2, three runs. Output is at vertex of connecting lines. Note a single run (solid connecting lines) shows noticeable positional error. a,b labels on inputs denote position in +(a,b) function.

2. Materials and Methods

2.1. Algorithm details

We set up a workspace with 2 sets of three input transducers, representing the numbers 0, 1 and 2. We selected three different addition relationships to encode into our PWC setup. We attempted to encode [+(0,0) = 0] by itself. We then added a new relation so we had [+(0,0) = 0 and +(1,1) =2]. Finally we added a one more algorithm for encoding, to obtain [+(0,0) = 0, +(1,1) = 2, +(1,2) = 3]. We termed an encoding successful when, given a particular set of selected input variables x and y for +(x,y) we were able to find a spatial arrangement such that the wavefronts from the emitting transducers representing the input variables possessed an intersection at a common location. For simple pairs of inputs this is straightforward. However as one attempts to encode multiple relationships, it becomes necessary to insure that wavefronts emitted from +(x,y) do not also intersect at the point of intersection for the output of +(x',y') where x' and y' are two other numbers to be added. This task is made much simpler by the fact that intersections between circular wavefronts emitted from a point lie along parabolic curves with the emitters as their foci, as such there are closed form solutions for finding these intersections. Additionally any real transponder would have some kind of spatial extent or form factor and detection of wavefront intersections would necessarily entail some ambiguity (for example, if an intersection detected when a wavefront crosses the physical boundary of

1.2. Polychronous Wavefront Computation: Rationale

Like artificial neural networks and spiking neural networks PWC architectures are universal Turing computers. Any algorithm that a Turing machine can encode can be encoded via a PWC architecture. However, it is potentially difficult to understand the means by which a PWC might encode an algorithm. Given this, let us consider a very simple algorithm: the addition of two integers.

A PWC binary adder can be instantiated in any number of ways. Functionally speaking, we represent this algorithm +(x,y). We propose a fairly simple implementation with input transponders assigned to the function inputs x and y, and an output transponder connected to a readout. If we are adding all pairs of numbers from 1 to 10, then, in this formulation we will need twenty input nodes (two arrays of nodes labeled x=0,x=1, x=2, x=3, .through x=9 and y =0,y=1, y=2, y=3…y=9) . For this range of inputs, then each potential output node would be labeled from o = 0 to o = 18. We illustrate simple subset of values for this adder in figure 1.

the transponder edge, a larger transponder would detect intersections a bit earlier than a smaller transponder). Figure 1 depicts encodings of +(1,1) for three distinct runs (figure 2). Note the positional error in the encoding of the output note at the vertex of the solid lines. This error results from a tolerance inserted into the models coincidence detection designed to account for a non-zero radius for each transponder.

We compared two algorithms for rapidly identifying transponder placement in each scenario: particle swarm optimization (PSO) and a genetic algorithm (GA). Particle swarm optimization algorithm is an algorithm which models search in some parameter space by modeling the behavior of a moving flock of agents, each representing the selection of some coordinate set in that space. Because there is a straightforward geometric interpretation for programming a polychromous wavefront computer (i.e., moving transponders around the workspace to find the correct configuration to represent an algorithm), we felt that using PSO to explore the positions in a real geometric space. As a point of comparison, we also chose to explore the performance of a generic genetic algorithm at this task. We performed runs for each optimization algorithm for each of the following three encodings:

- +(0,0) = 0
- ,+(0,0) = 0 and +(1,1) =2
- +(0,0) = 0, +(1,1) = 2, +(1,2) = 3

For both particle swarm and genetic algorithms we chose to run 10 generations with 10 replicates apiece. We ran each optimization 5 times. We quantified the performance of each optimization algorithm as a *compiler* for a PWC algorithm by tracking both the total time for convergence of all simulations, as well as the corresponding total positional error (TPE) value (deviation of the estimated transponder positions from the closest points on the corresponding parabolae of intersection), a measure of the transponder "form factor" required for detection of signal coincidence.

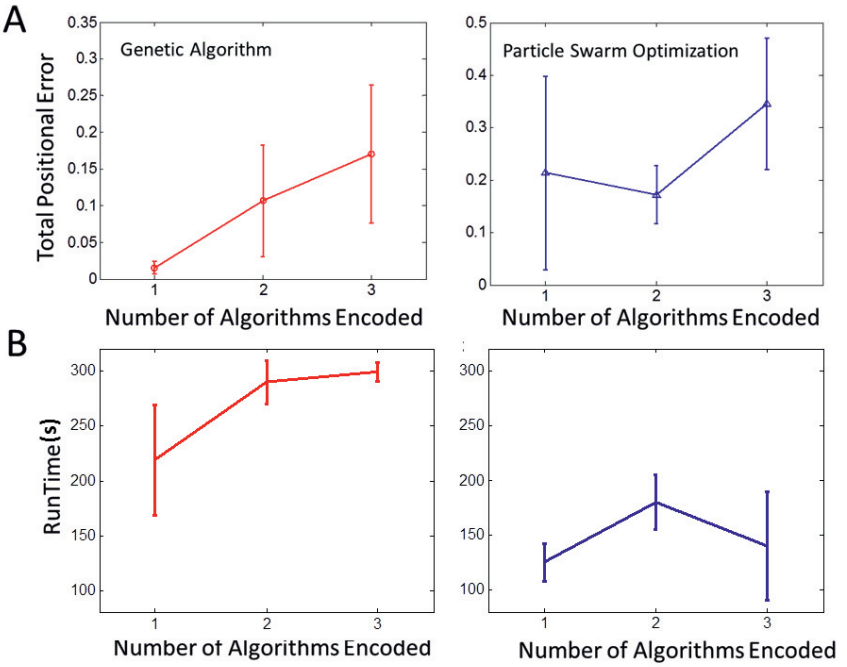

Figure 3 A) Genetic algorithmsignificantly outperform particle swarm optimization for encoding a single algorithm. Performance of GA rapidly degrades with increasing number of algorithms stored to be indistinguishable from PSO approach. PSO show no clear linear trend. B) PSO based encoding shows a marked advantage in terms of time.

3. Results

3.1. Comparing Particle Swarm and Genetic Algorithm swarm results.

Comparing the two optimization schemes, results were somewhat mixed. While the overall margin-of-error (TPE) in terms of transponder placement was lower for the genetic algorithm optimization for learning of only 1 or 2 +(x,y) algorithm encodings, TPE showed a clear linear and significant increase with the number of algorithms encoded. In contrast particle swarm optimization showed overall higher encoding TPEs, but showed no indication of an increasing trend. Furthermore for encodings of 2 and 3 algorithms the TPE was somewhat higher, but not *significantly* so for PSO (figure 3A).

The run time for each form of optimization showed a much clearer picture (figure 3B). In general, "compiling" a PWC algorithm into geometry took significantly less time for PSO for each cardinality of algorithms to be encoded into the PWC processor than for GA optimization. Indeed, for the PSO approach, encoding times were half or less those seen in the GA experiments. While for a single algorithm encoding [+(0,0) = 0], this is still problematic, as the TPE for this encoding is significantly lower for GA algorithms for the two algorithm and three algorithm encodings it represents a potential advantage for PSO, given that TPE in these cases did not *significantly* differ between the two approaches.

4. Discussion

Examining the results of this simple experiment we found a somewhat complicated picture. While neither algorithm appeared to be a clear winner with respect to acting as a method to "compile" programs for polychromous wavefront computers, each appeared to have some desirable features. For the very small numbers of algorithms encoded in this experiment, genetic algorithms appeared to find arrangements with smaller TPEs for simultaneous signal detection, and as such, may allow tighter packing of transponders in a physical instantiation of the program. However the same data indicates that GA-derived encodings yield a linear increase in TPE with the number of encoded algorithms. This increase in error with the number of encoded algorithms is not evident in particle swarm-based optimization, which may imply that, for large numbers of algorithms, a geometric approach (as in PSO) would be preferable. Convergence times for PWC programming via particle swarm optimization were significantly smaller than genetic algorithm based programming, which seems to support to the notion that aligning a programming methodology with the geometry of the encoding problem is important. So while the situation is mixed, these results (a flat relationship between the number of algortithms to be encoded and positional error, and shorter convergence times) appear to indicate that programming by optimization must consider the geometry in which transponders are to be embedded in order to be successful. Given that points of intersection between two wavefronts lie on parabolae with the origin of the wavesfronts as foci, perhaps projecting the transponders into a parabolic coordinate system and performing the optimization there might prove fruitful.

While this is very preliminary work, it represents the early stages of what we hope to be a sustained investigation into novel non-von Neumann computational architectures, specifically architectures that exploit temporal encodings and time delays between events as key computational elements. Such architectures potentially offer huge computational efficiencies[4] and would have application in domains such as event correlation, change detection, or even predictive analytics. Ultimately, we hope to be able to be able to use this information to identify a framework of best practices for developing and programming polychromous computer hardware.

References

1. Simon, D.R, "On the power of quantum computation".*Foundations of Computer Science, 1994 Proceedings., 35th Annual Symposium on*: 1994. p116–123.
2. Abraham, E., Seaton, C., and Smith, S., (1983) The optical computer. *Scientific American*, pages 63—71
3. Arecchi, F., , Boccaletti, S., Ducci, S., Pampaloni, E., Ramazza, P., Residori, S., (2000) The Liquid Crystal Light Valve with Optical Feedback: A Case Study In Pattern Formation Journal Of Nonlinear Optical Physics & Materials vol. 9, no. 2 183{204)
4. Venkataramani, S., Chippa, V.K., Chakradhar,S.,T, Roy,S., Raghunathan, A,. Quality programmable vector processors for approximate computing. Proceedings of the 46th Annual IEEE/ACM International Symposium on Microarchitecture 2013. p1-12
5. Monroe, D. "Neuromorphic computing gets ready for the (really) big time". *Communications of the ACM* 2014. 57 (6): 13–15
6. Painkras E., Plana, L.A., Garside, J., Temple, S., Davidson, S., Pepper, J., Clark, D., Patterson, C., and Furber, S., SpiNNaker: A Multi-Core System-on-Chip for Massively-Parallel Neural Net Simulation 2012 IEEE Custom Integrated Circuits Conference (CICC) 2012
7. Maass, W., NeuralNetworks, Networks of Spiking Neurons: The Third Generation of Neural Network Models 1997. 10:1659-1671 1
8. Izhikevich, E.M., Hoppensteadt, F.C. : Polychronous Wavefront Computations. I. J. Bifurcation and Chaos 2009. 19(5): 1733-1739
9. Wu, K., García de Abajo, J.,Soci,, C., Shum, P., Zheludev, N., (2014) An optical fiber network oracle for NP-complete problems. *Light: Science & Applications* 3, e147; doi:10.1038/lsa.2014.28
10. Yang, L.,, Zhang, L., and Ji, R., , (2013) On-chip optical matrix-vector multiplier for parallel computation, SPIE 10.1117/2.1201306.004932